VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Vzdálená správa komunikačního systému pomoci webových služeb

Remote Management of Communication Systems through Web Service

2012 **Marián Gryga**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Marián Gryga**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Vzdálená správa komunikačního systému pomocí webových služeb
Remote Management of Communication System through Web Services

Zásady pro vypracování:

Cílem bakalářské práce je naprogramovat vzdálenou správu komunikačního systému pomocí webových služeb, založených na protokolu SOAP. Jako komunikační systém bude použit Asterisk.
1. Úvod do problematiky webových služeb.
2. Implementace webových služeb pro vzdálenou správu komunikačního systému.
3. Testování a ověření funkčnosti s koncovými zařízeními.

Seznam doporučené odborné literatury:

Chappell D. A., Jewell T. *Java Web Services* O'Reilly Media 2002
Meggelen J. V. *Asterisk: The Future of Telephony* O'Reilly Media 2007
Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Nevlud**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne:1.5.2012

Poděkování

Rád bych poděkoval Ing. Pavlu Nevludovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Bakalářská práce se zabývá vzdálenou správou Asterisk pobočkové softwarové telefonní ústředny pomocí webové služby. Klient komunikuje s touto službou za pomocí SOAP protokolu. Služba je vyvíjena v programovacím jazyce Java a používá Asterisk Java rozhraní. Hlavním učelem služby je provádění AMI příkazů. Tato služba je nasazena na aplikační server GlassFish. Vše pracuje v operačním systému Linux.

Klíčová slova

webová služba, komunikační systém, Asterisk, AMI

Abstract

This bachelor thesis deals with the remote management of Asterisk private branch exchange using Web service. The client of service communicates with the service using SOAP protocol. The service is developed in Java programming language and uses the Asterisk Java interface. The main objective is to perform AMI commands. This service is deployed on GlassFish application server. All is running under the Linux operating system.

Key words

Web Service, communication system, Asterisk, AMI

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
IT	Information Technology	Informační technologie
WS	Web Services	Webové služby
TDM	Time Divided Multiplexing	Časově dělený multiplex
VoIP	Voice over Internet Protocol	Internetová telefonie
RPC	Remote Procedure Call	Vzdálené volání procedur
SIP	Session Initiation Protocol	Protokol pro inicializaci relací
RMI	Java remote method invocation	Java volání vzdálených metod
EDI	Electronic Data Interchange	Elektronická výměna dat
WDDX	Web Distributed Data Exchange	Výměna distribuovaných dat
HTTP	Hyper Transfer Protocol	Internetový protokol
SMTP	Simple Mail Transfer Protocol	Protokol elektronické pošty
TCP/IP	Transmission Control / Internet Protocols	Protokoly pro komunikaci v síti
FTP	File Transfer Protocol	Protokol pro přenos souborů
Java EE	Java Enterprice Edition¨	Rozšíření jazyka Java
XML	eXtensible Markup Language	Rozšiřitelný značkovací jazyk
PSTN	Public switched telephone network	Veřejná komutovaná telefonní síť
PBX	Private Branch eXchange	Přepínání soukromých větví
IVR	Interactive Voice Response	Interaktivní hlasová odezva
API	Application Programming Interface	Programové rozhraní aplikace
URI	Uniform Resource Identifier	Jednotný identifikátor zdroje
JAX-WS	Java API for XML web services	Java API pro XML webové služby
URL	Uniform Resource Locators	Jednotný popis umístění zdroje
URI	Uniform Resource Identifier	Univerzální identifikátor zdroje
CSS	Cascading Style Sheets	Kaskádové styly

W3C	Word Wide Web consortium
REST	Representational State Transfer
SOAP	Simple Object Access ProtocolWeb Services
WSDL	Web Services Description Language
UDDI	Universal Description Discovery and Integration
MGCP	Media Gateway Control Protocol
IAX	Inter-Asterisk eXchange
JRE	Java Runtime Environment
OSS	Open Sound System
ALSA	Advanced Linux Sound Architecture
EJB	Enterprise Java Beans
AMI	Asterisk manager API
CORBA	Comon Object Request Broke Architecture
DCOM	Distributed Component Object Model
JSP	Java Server Pages
POTS	Plain Old Telephone Service
H.323	
E1	
T1	

Obsah

1 Úvod

Webové služby. Tento pojem začal svět počítačových sítí a internetu vnímat příchodem nového milénia. Na samotném počátku stály velké firmy jako Microsoft, IBM, SUN. Základní myšlenkou technologie nazývané webové služby (Web Services) byla standardizace dřívějších technologií pro vzdálené volání funkcí známé jako COBRA, DCOM, RPC, či RMI. Webové služby nabízí spolupráci počítačů v síti jak na Internetu, tak v Intranetu, za použití standardních protokolů nejčastěji HTTP, SMTP, FTP a dalších. Jsou nezávislé na platformě a existuje mnoho implementací těchto služeb pro různé programovací jazyky např. C/C++, Java, .NET a další. Aplikace využívající webových služeb si mezi sebou zasílají XML zprávy, ve kterých jsou přenášeny dotazy a odpovědi specifické pro danou službu.

Mým cílem bylo vytvořit webovou službu pro vzdálenou správu komunikačního systému Asterisk v operačním systému Linux. V aplikaci byl použít programovací jazyce Java, využívající Java EE, ve vývojovém prostředí NetBeans za pomocí JAX-WS. Tato aplikace je nasazena na aplikační server GlassFish, testována prostřednictvím volně dostupné verze software soupUI a jednoduchého webového klienta. Javu jsem si vybral z důvodu předchozích studijních zkušeností.

Kapitola 2 popisuje teoretický úvod do problematiky webových služeb. Obsahuje základní principy webových služeb, komunikaci mezi poskytovatelem služby a klientem služby, protokoly pro komunikaci, které jsou nezbytné pro seznámení čtenáře s webovými službami a lepší orientaci v navazujících kapitolách.

Kapitola 3 se zabývá vlastní implementací webové služby a implementací klienta této služby pro její ovládání.

Kapitola 4 se věnuje testování webové služby v programu SoapUI a klienta webové služby, na virtuálních Linux serverech v prostředí Oracle Virtual Box.

2 Úvod do problematiky webových služeb

2.1 Historie webových služeb

„Myslím si, že mohu bezpečně tvrdit, že Microsoft není první, kdo spojil tyto dvě slova dohromady, ale byli jsme první, kdo tyto dvě slova spojil s XML, SOAP, WSDL, UDDI“. Tyto slova vyřkl John Montgomery příchodem nového milénia. Avšak neoficiální důkazy naznačují, že spojení těchto slov použil poprvé tehdejší šéf Microsoftu Bill Gates na konferenci v Orlandu 12. července 2000.

2.1.1 Předchůdci webových služeb

Prvním pokusem o standard v komunikaci po síti byl bezesporu EDI spuštěný v roce 1975. Po několik desítek let od spuštění EDI bylo několik pokusů o universální cestu, jak propojit business logiku po síti.

Technologie jako:

- **CORBA**
- **DCOM**
- **RPC**
- **RMI**

Žádné z těchto technologií se nepodařilo získat významný podíl na trhu a vážnější úspěch, avšak většina se používá i dnes. EDI nebylo jednoduché implementovat pro jeho složitost a nákladnost. CORBA byla nasazována na Unix platformě a DCOM na Microsoft platformě. Obě tyto platformy mezi sebou mnoho let soupeřili. Všem těmto technologiím chyběla podpora významných IT společností. Očekával se tedy úspěch webových služeb.

2.1.2 Prostředí webových služeb

Už z názvu je patrné, že webové služby jsou primárně určeny pro spolupráci v síti. V síti je používán HTTP internetový protokol, přenášený TCP/IP protokoly, jenž byl v roce 1997 přijat jako univerzální standart. Bylo však zapotřebí najít způsob pro zapouzdření zpráv a dat jako standart, používaný v mnoha IT společnostech. V roce 1998 právě XML jako standard začalo budovat správnou cestu pro webové služby. Allaire Corp.'s WDDX byl mezi prvními pokusy o úspěch, byl to však SOAP vyvíjený ve spolupráci z Davem Winerem, Bobem Atkinsonem, Donem Boxem a Mohsen Al-Ghoseinem.

2.1.3 Webové služby jako standard

V roce 1999 uspořádal Microsoft společně s DevelopMentor a dalšími zainteresovanými schůzku, kde byl předveden SOAP verze 1.0, specifikace pro zasílání zpráv standardními protokoly, postavený na XML.

Na začátku roku 2000 IBM veřejně podpořil Microsoft a společně s zainteresovanými IT společnostmi se podíleli na vývoji SOAP verze 1.1. V polovině tohoto roku vydávají specifikaci WSDL. Spojením těchto technologií dostávají IT společnosti do rukou nástroj pro vytváření a popis webových služeb. Je však nutno nalézt způsob, jak tyto služby propagovat a lokalizovat. Proto ke konci roku 2000 vydávají UDDI verze 1.0. Spojením těchto technologií byl položen základ pro standard webových služeb. Avšak až koncem roku 2000 se významné společnosti jako Oracle, HP, Sun, IBM, a Microsoft dohodly podporovat standard webových služeb ve svých produktech. Éra webových služeb byla odstartována. Začátkem roku 2002 se pojem webové služby stává standardem konsorcia W3C a jejich vývoj, na kterém se podílelo mnoho významných IT společností, pokračuje.

2.2 Webové služby

Webové služby (Web Services), jak již samotný název napovídá, jsou služby běžící v prostředí webového API, na síti. Můžeme si je představit jako služby umožňující komunikaci systémům, bez ohledu na to v jakém jazyce jsou naprogramovány a na jaké platformě aktuálně běží. Komunikace mezi těmito službami probíhá zpravidla pomocí XML zpráv a protokolu HTTP. Webové služby jako většina API jsou definovány jako standard. Služby se dělí na základní dva tábory SOAP a REST webové služby. Dle volného překladu tento standard W3C definuje jako.

„Webová služba je softwarový systém navržený k efektivní spolupráci mezi stroji přes síť. Má rozhraní popsané ve strojově srozumitelném formátu konkrétně WSDL. Okolní systémy s webovou službou komunikují stanoveným způsobem pomocí SOAP zpráv, přenášených obvykle HTTP protokolem s XML serializací ve spojení z ostatními webovými standardy“.

Stavební kameny webových služeb:

- **XML**
- **SOAP**
- **WSDL**
- **UDDI**

Tyto technologie jsou popsány v následujících kapitolách.

2.2.1 Přenos webové služby

Webové služby používají pro svůj přenos zpravidla protokol HTTP a pro výměnu dat obecný značkovací jazyk XML. V následující podkapitole se budu stručně zabývat právě tímto protokolem.

2.2.1.1 HTTP protokol

Hypertext transport protokol je internetový protokol primárně navržený pro přenos hypertextových dokumentů ve formátu HTML. Obvykle používá TCP port 80. Nejpoužívanější protokol v dnešním internetu. Používá se však i pro přenos dalších informací. Společně s XML tvoří základ pro přenos Webových služeb. Protokol komunikuje na bázi dotaz-odpověď. Uživatel zašle serveru dotaz a server následně na dotaz odpoví. Jedná se o bezstavovou komunikaci, proto server neudržuje žádný stav mezí odesíláním požadavků. HTTP definuje určitá pravidla, jak má vypadat formát komunikace mezi klientem a serverem a požívá několik dotazovacích metod a stavových kódů, které tato komunikace vyžaduje. Zmíním se však pouze o metodách GET a POST, protože zejména metoda POST společně s XML je používána k přenosu webových služeb. Více informací o HTTP protokolu, jeho metodách a stavových kódech naleznete zde [01].

2.2.1.2 Metoda GET

Jedná se o metodu, která přenáší data prostřednictvím URL adresy za otazníkem. Proto mohou být tyto data uživatelem lehce změněna přepsáním URL. Metoda je proto používaná především pro přenos bezpečných dat např. předáním vyhledávaného řetězce.

2.2.1.3 Metoda POST

Odesílá data uživatele na server. Používá se pro zasílání většího objemu dat a data se přenáší jako součást těla dotazu, proto nejsou v URL viditelná. Převážně se tato metoda používá pro odesílání formulářů na server a zasílání choulostivých dat např. hesel. Spojením této metody s XML ideálně vyhovuje k přenosu služby.

2.2.1.4 Ukázka HTTP komunikace

Dotaz klienta:

```
POST /AsteriskWebService HTTP/1.1
Accept: text/xml, multipart/related
Content-Type: text/xml; charset=utf-8
Host: localhost:8080
Connection: keep-alive
Content-Length: 282
```

Odpověď serveru na dotaz:

```
HTTP/1.1 200 OK
server: grizzly/1.9.36
Content-Type: text/xml;charset=utf-8
Transfer-Encoding: chunked
Date: Sun, 22 Apr 2012 11:13:19 GMT
```

2.2.1.5 XML

XML (rozšiřitelný značkovací jazyk) je standardem W3C konsorcia. Dokument v XML má stromovou strukturu s jedním kořenem, uzly stromu jako elementy a listy stromu jako elementy, atributy nebo text. Jedná se o jakýsi soubor pravidel tvorby textových formátů, které uspořádávají data v XML strukturách. Usnadňují tvorbu, čtení a zápis dat počítačem. Tento jazyk je proto rozšířitelný, nezávislý na platformě a podporuje lokalizaci. Plně vyhovuje standardu Unicode.

2.2.1.6 Struktura XML dokumentu

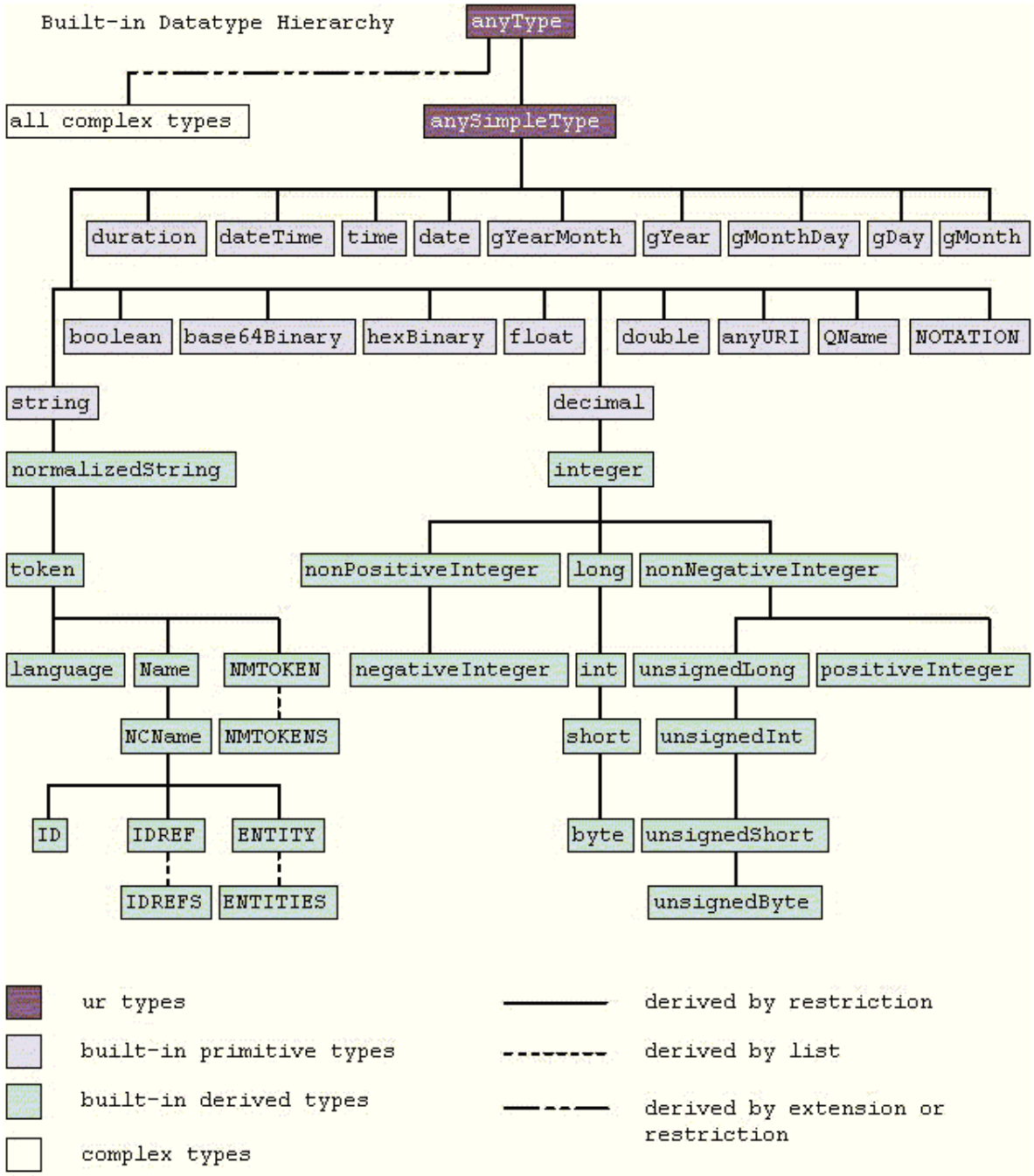
- **Hlavička** - deklaruje XML dokument. Je v dokumentu nepovinná, avšak pokud je uvedena musí být na prvním řádku. V hlavičce je uvedena verze XML, dále pak kódování dokumentu atd.
- **Elementy** - vyznačují se v textu pomocí tzv. *tagů*. Element se skládá z počátečního a koncového tagu. Dokument musí obsahovat jeden kořenový element.
- **Atributy** - doplňující informace k elementům. Element může obsahovat více atributů. Atributy zapisujeme do počátečního elementu ve tvaru: jméno atributu, rovná se a hodnota atributu v uvozovkách nebo apostrofech.
- **Znakové entity** - pro zápis některých znaků, např. „<“, které se používají k oddělení tagů od textu v dokumentu, zavádí XML tzv. *znakové entity*.
- **Komentáře** - slouží k vysvětlení či skrytí části dokumentu.

2.2.1.7 XML jmenné prostory

Jmenný prostor je sada jmen, jenž můžeme opakovaně v XML dokumentu používat. Obsahuje elementy a atributy. Sada těchto jmen je identifikována pomocí URI. Názvy jmenného prostoru se skládají z prefixu, dvojtečky a lokální části.

2.2.1.8 *XML schémata*

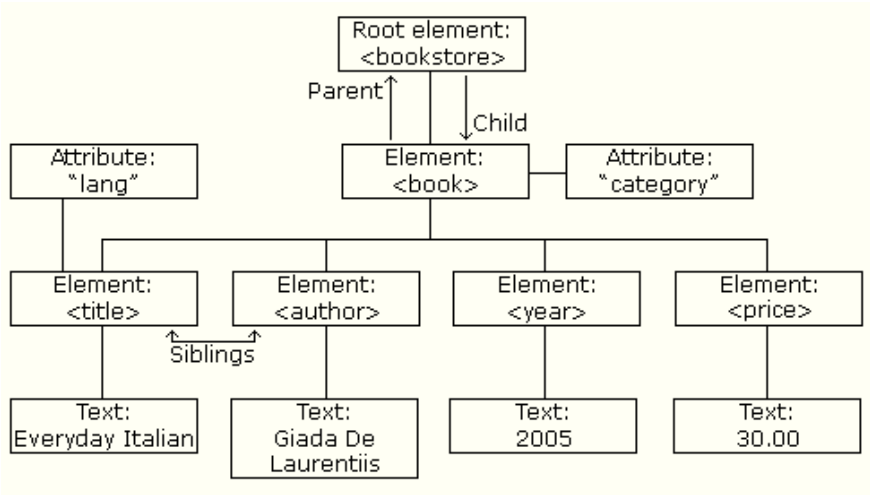
XML je definicí normy datových typů a datových struktur v XML dokumentu.



Obrázek 2.1 Datové typy v XML schématu. Převzato z [01]

Na obrázku 2.1 vidíme uspořádání všech datových typů. Tyto typy se dělí do dvou základních skupin, jednoduché typy (simple type) a složené typy (complex type). Více informací a podrobný popis jazyka XML naleznete zde [02].

2.2.1.9 *Ukázka XML dokumentu*



Obrázek 2.2. Struktura XML dokumentu. Převzato z [03]

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Bookstore XML document-->
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```

Ukázka definice jmenného prostoru:

```
<schemaxmlns="http://www.w3.org/2001/XMLSchema"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://example.com/person" >
<complexTypename="Person">
  <sequence>
    <element name="name" type="xsd:string"/>
    <element name="address" type="xsd:string"/>
  </sequence>
</complexType>
```

2.2.2 SOAP

SOAP je protokol pro výměnu informací v distribuovaných systémech pracující na principu vzdáleného volání procedur RPC. Definuje strukturu zprávy, které si v distribuovaných prostředích aplikace zasílají. Struktura zprávy je ve formátu XML. Jedná se o protokol aplikační vrstvy, který pro přenos zprávy používá jiné protokoly této vrstvy, zpravidla HTTP.

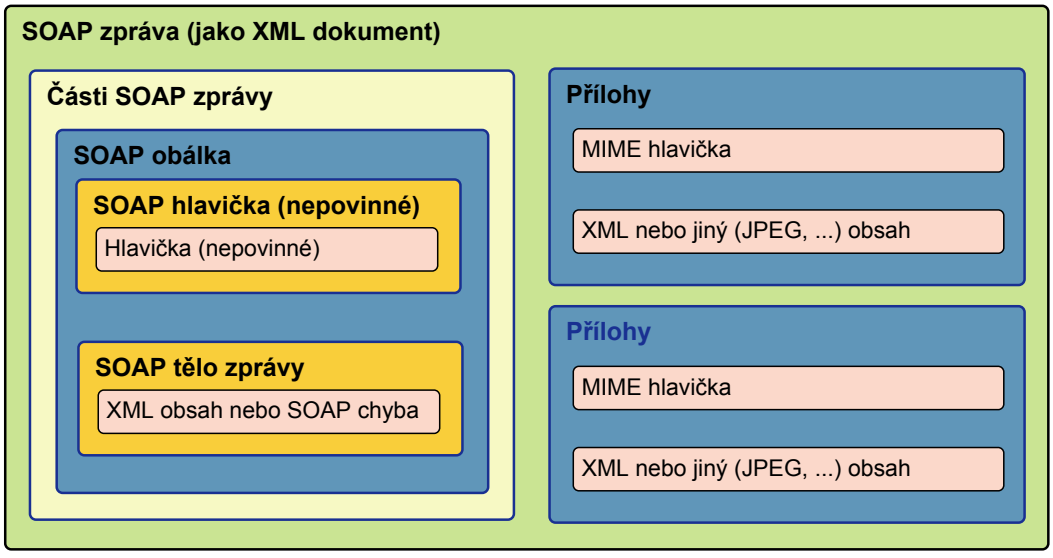
2.2.2.1 Formát zprávy

XML definuje základ formátu SOAP zprávy, pro jeho rozšiřitelnost a podporu vývojových nástrojů. XML však sebou přináší výhody v čitelnosti zprávy uživatelem a v možnosti validovat obsah ve zprávě. Ale i nevýhody v podobě vysoké režie při pársování počítačem v závislosti na množství přenesených dat a tedy pomalejší komunikaci oproti ostatním distribuovaným systémům komunikujícím prostřednictvím výměny binárních dat.

2.2.2.2 Tři části SAOP zprávy

SOAP zpráva se skládá z obálky, hlavičky a těla zprávy.

- **Obálka** (envelope) - kořenový element zprávy. Definuje jmenné prostory (namespaces) a obsahuje další elementy hlavičku a tělo.
- **Hlavička** (header) - prvek uvnitř obálky, nepovinný prvek obsahující informace specifické pro damou aplikaci, například autentizaci, transakce a další.
- **Tělo** (body) - prvek uvnitř obálky, informuje server kterou z metod má provést. V případě odpovědi, výsledek požadované operace.



Obrázek 2.3. Struktura SOAP zprávy.

2.2.2.3 Druh volání služeb v SOAP

- **Jednosměrná** (one way) - volání s parametry, nevrací hodnotu.
- **Oznamovací** (notification) - volání bez parametru, vrací hodnotu.
- **Dotaz - Odpověď** (request - reaponse) - volání s parametrem, vrací hodnotu.

2.2.2.4 Ukázka SOAP komunikace

Tato ukázka reprezentuje jednu z možností komunikaci mezi klientem a serverem. Volá funkci pro převod teploty ze stupňů Fahrenheita na stupně Celsia.

Dotaz klienta volající metodu na serveru:

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soapenvelope/">
  <S:Body>
    <ns2:farenheitToCelsius xmlns:ns2="http://webservice.vsb.cz/">
      <farenheit>0.0</farenheit>
    </ns2:farenheitToCelsius>
  </S:Body>
</S:Envelope>
```

Na tomto jednoduchém příkladu SOAP dotazu na server, vidíme element `envelope`, který definuje základní jmenný prostor pro protokol SOAP. Element reprezentující hlavičku ve zprávě není definován. V těle zprávy máme element s názvem funkce `farenheitToCelsius`, která převede parametr specifikovaný v elementu `farenheit`.

Odpověď serveru:

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:farenheitToCelsiusResponse
     xmlns:ns2="http://webservice.vsb.cz/">
      <return>-32.0</return>
    </ns2:farenheitToCelsiusResponse>
  </S:Body>
</S:Envelope>
```

V odpovědi vidíme výsledek volané funkce `farenheitToCelsius`.

Na nesledujícím jednoduchém příkladu můžeme vidět komunikaci mezi klientem a serverem pomocí SOAP zprávy. Tato komunikace může být uskutečněna pomocí některého s protokolů aplikační vrstvy, nejčastěji však HTTP. Existují i velice složíte typy SOAP zpráv, avšak pro pochopení základního principu SOAP je tento příklad dostačující. Více informací o protokolu SOAP naleznete zde [04].

2.2.3 WSDL

WSDL jazyk, je primárně určen k popisu webových služeb. Jedná se o množinu koncových bodů (service endpoints). Tento koncový bod si můžeme představit jako rozhraní dané služby. Stejně jako SOAP je tento jazyk postaven na XML. Nejčastěji se používá právě ve spojení s XML schématem a protokolem SOAP. Jazyk popisuje operace konkrétních služeb, také data, které WSDL používá pro volání jednotlivých operací. Dále adresy a porty pro komunikaci s danou službou. WSDL je standard W3C konsorcia a aktuální verze je WSDL 2.0. Tento dokument je složen ze dvou částí a to jsou abstraktní a konkrétní.

2.2.3.1 Abstraktní popis

Abstraktní popis definuje rozhraní služby, pro odesílání a příjem zpráv a poskytovaných operací, bez ohledu na technologii jenž je služba implementována či operačním systéme na kterém služba běží a protokolu který služba používá.

Abstraktní popis a jeho tři základní oddíly.

- **Rozhraní** (interface) - rozhraní služby, jaké operace služba poskytuje.
- **Operace** (operation) - popisuje operace a jejich vstupní a výstupní parametry.
- **Zprávy** (message) - popisuje zprávy představující operace a jejich parametry.

2.2.3.2 Konkrétní popis služby

Navazuje abstraktní popis na reálnou implementaci a komunikaci na konkrétní protokol.

Konkrétní popis a jeho tři základní oddíly.

- **Spojení** (binding) - popisuje požadavek služby pro navázání skutečného spojení, pro operace či celý interface.
- **Služba** (service) - seskupuje prvky endpoint, tím popisuje službu.
- **Výstupní bod** (endpoint) - vymezuje adresu, na které bude služba přístupná.

2.2.3.3 Ukázka WSDL

Definice jmenných prostoru v dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl"
             xmlns:tns="http://www.tmsws.com/wsdl20sample"
             xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:wsoap="http://schemas.xmlsoap.org/wsdl/soap/"
     targetNamespace=" http://webservice.vsb.cz/"
     name="TemperatureConvertor">

<types>
  <xsd:schema>
    <xsd:import namespace="http://webservice.vsb.cz/"
     schemaLocation="http://localhost:8080/TemperatureConvertorWS/Te
     mperatureConvertor?xsd=1"/>
  </xsd:schema>
</types>
```

Definice zprávy pro danou službou:

```
<message name="farenheitToCelsius">
  <part name="parameters" element="tns:farenheitToCelsius"/>
</message>
<message name="farenheitToCelsiusResponse">
  <part name="parameters" element="tns:farenheitToCelsiusResponse"/>
</message>
<portType name="TemperatureConvertor">
  <operation name="farenheitToCelsius">
    <input
     wsam:Action="http://webservice.vsb.cz/TemperatureConvertor/fare
nheitToCelsiusRequest" message="tns:farenheitToCelsius"/>
    <output
     wsam:Action="http://webservice.vsb.cz/TemperatureConvertor/fare
nheitToCelsiusResponse" message="tns:farenheitToCelsiusResponse"/>
  </operation>
</portType>
```

Formát volání operací a umístění služby:

```
<binding name="TemperatureConvertorPortBinding"
     type="tns:TemperatureConvertor">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
     style="document"/>
  <operation name="farenheitToCelsius">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>


</definitions>
```

Následující příklad nám ukazuje strukturu WSDL dokumentu jednoduché webové služby. XML schéma bylo však pro samotný popis nevyhovující. WSDL 2.0 proto zavedla model komponent (component model), jenž definuje komponenty, které dohromady popisují jednotlivé webové služby. V tomto příkladu jsou některé z těchto komponent použity. Podrobný popis všech komponent WSDL 2.0 naleznete zde [04].

2.2.4 UDDI

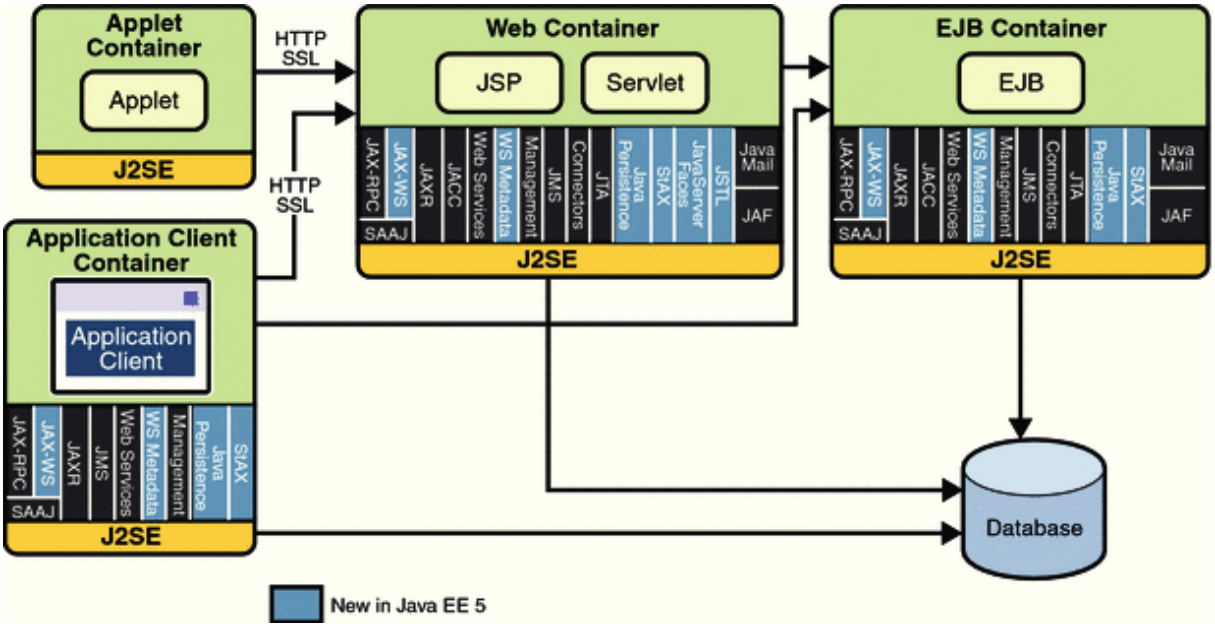
UDDI je mechanizmus pro registraci, kategorizaci a vyhledávání webových služeb. Jedná se o další webovou službu, která funguje jako velký adresář, ve kterém můžeme nalézt informace o subjektech (firmách) a poskytovaných službách. V tomto adresáři můžeme vyhledávat podle různě zadaných kriterií. Tento registr byl plně veřejný, což vedlo k faktu, že většina záznamů byla nepoužitelná a špatná. Proto v lednu 2006 hlavní provozovatelé IBM, SAP a Microsoft tyto své rejstříky vypnuli. Dnešní použití UDDI nalezneme pouze ve vnitropodnikové struktuře.

2.2.5 REST

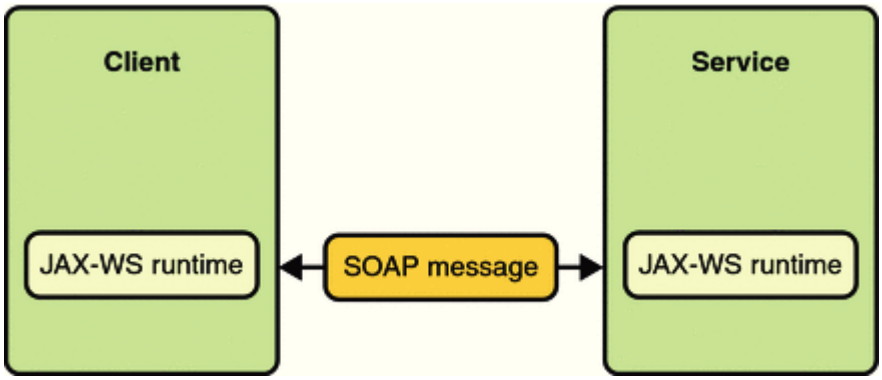
Další technologií webových služeb je Representation State Transfer (REST). Jedná se o architekturu rozhraní pro prostředí distribuovaných systémů. REST by navržen v roce 2000 Royem Fieldingem v rámci jeho disertační práce. Toto rozhraní umožňuje jednoduchý a snadný přístup ke zdrojům. Těmito zdroji mohou být data nebo stavy aplikace. Proto je REST oproti SOAP orientován datově. Tato technologie využívá metod protokolu HTTP (POST, PUT, DELETE, ...) a tím předává data službě. V této práci se však REST webovými službami nebudu zabývat, proto je zmiňuju jen okrajově. Vice informací o REST webových službách naleznete zde [05].

2.3 JAX-WS

JAX-WS jedna z řady technologií Java EE používaných pro vývoj SOAP webových služeb a jejich klientů. Toto API bylo vyvinuto jako následník JAX-RPC. Hlavním motivem tohoto API je poskytnout rozhraní pro ulehčení vývoje webových služeb, zejména pak oddělit programátora od SOAP a WSDL zpráv. Hlavním přínosem je přechod na Java 5 a používání anotací. Podporují „message-oriented“ a „RPC-oriented“ volání webových služeb. Zjednodušuje práci s koncovým bodem. Webové služby jsou definovány jako Java třída s metodami označenými anotacemi. Specifikace webových služeb JAX-WS respektuje nezávislost na platformě.

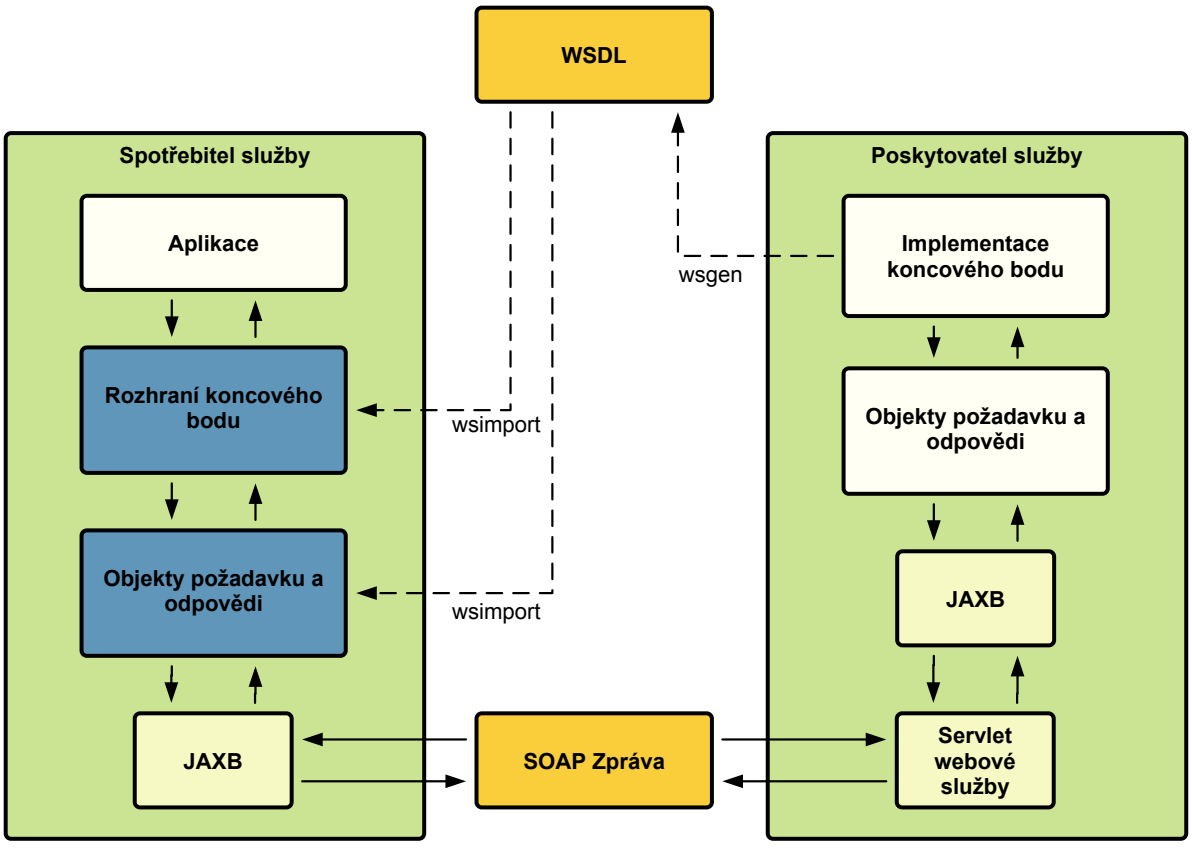


Obrázek 2.4 Technologie JAX-WS v Java EE. Převzato z [06]



Obrázek 2.5 JAX-WS komunikace pomocí SOAP. Převzato z [06]

- **Poskytovatel služby** (services provider) - vytvoří webové služby jako Java třídy s metodami, které daná služba poskytuje označenými pomocí anotací. Popis WSDL je generován automaticky dle definice anotovaného rozhraní třídy, stejně jako obslužný kód navazující tuto třídu na Java EE API .
- **Spotřebitel služby** (services consumer) - vytvoří lokální proxy pro volání webové služby a tu využívá. Proxy využívá stejné rozhraní jako třída implementující webovou službu u poskytovatelé.



Obrázek 2.6. Spolupráce spotřebitele a poskytovatele v JAX-WS.

Na obrázku 2.7 můžeme vidět, jak probíhá komunikace mezi spotřebitelem a poskytovatelem webové služby. Dále zde vidíme barevně rozlišené části, kde bílé časti implementuje programátor, modré části jsou automaticky generovaný kód a žluté části poskytuje JAX-WS API a další související knihovny Java EE API.

Programátor implementuje koncový bod a specifikuje objekty požadavku a odpovědi.Z nich je pak schopen pomocí `wsgen` příkazu vygenerovat WSDL dokument reprezentující rozhraní pro danou službu. Programátor na straně klienta použije příkaz `wsimport`, který vytvoří rozhraní koncového bodu poskytované služby a objekty pro komunikaci s touto službou.

2.3.1.1 Ukázka jednoduché JAX-WS služby

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(serviceName = "WebSerivice")
public class WebSerivice {

    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

Na následujícím příkladu vidíme jednoduchou webovou službu. Implementace koncového bodu v této třídě představuje anotace `@WebService` a je tedy poměrně jednoduchá. Dále stačí jen definovat metody v této třídě a ty musí být opatřeny anotací `@WebMethod`, čímž je určen název operace webové služby. Parametry dané metody anotujeme pomocí @WebParam, jenž určuje jméno parametru služby. Nyní už jen stačí vygenerovat WSDL dokument příkazem `wsgen` a službu společně s WSDL nasadit na aplikační server. V předchozí kapitole jsem popsal stručný úvod do JAX-WS webových služeb. Více informací o technologii JAX-WS naleznete zde [05].

3 Implementace webové služby pro vzdálenou zprávu komunikačního systému

3.1 Současný stav

V současné době neexistuje plnohodnotná webová služba, která by poskytovala nástroj pro komplexní vzdálenou zprávu komunikačního systému Asterisk a poskytovala veškeré funkce nabízené Asterisk manager API [07]. Existuje však alfa verze Asterisk SOAP webové služby [], která se snaží vytvořit plnohodnotné ovládání Asterisk PBX pomoci webové služby a protokolu SOAP. Tato služba však v současné době nabízí jen několik metod pro zprávu Asterisk PBX.

3.2 Analýza

Již ze zadání je patrné, že webová služba bude používat protokol SOAP pro komunikaci s klientem této služby. Bude použitá platforma Java EE a zejména JAX-WS pro snadnou implementaci dané služby. Pro komunikaci s telefonní ústřednou Asterisk bude použito Asterisk Java rozhraní a tato služba bude nasazena na aplikačním serveru GlassFish. Služba bude poskytovat základní metody rozhraní Asterisk Java, zejména pak metodu pro provádění příkazů v prostředí Asterisku.

Klient webové služby bude používat stejnou platformu jako webová služba, tedy Java EE a její komponenty, zejména Servlet a Java Server Pages (JSP) pro komunikaci s webovou službou. Tento klient bude nasazen na aplikačním serveru GlassFish, vedle dané webové služby a přístupný pro všechny dnes používané webové prohlížečích např. Mozila Firefox nebo Google Chrome.

3.3 Komunikační systém

3.3.1 Softwarová ústředna Asterisk

Asterisk je kompletní open source hybridní TDM a pocket voice PBX softwarová pobočková ústředna, běžící na Unix, Linux platformě. Tento systém je flexibilní a rozšiřitelné řešení telekomunikačního software pod licencí. Své uplatnění najde v celé řadě aplikací, od pobočkových ústředen, VoIP bran, Interaktivních hlasových průvodců a mnoho dalších. Celý systém byl navržen tak, aby vytvořil komplexní rozhraní telefonnímu hardware, software a libovolných telefonních aplikaci.

3.3.2 Použití Asterisku

Asterisk lze použít v mnoha aplikacích např:

- **Pobočková ústředna** (PBX)
- **Různorodá VoIP gateway** (MGCP, SIP, IAX, H.323)
- **Interaktivní hlasový průvodce** (IVR)
- **Voicemail služby s adresářem**
- **Softwarová ústředna** (Softswitch)
- **Packet voice server**
- **Konferenční server**
- **Šifrování telefonních nebo faxových volání**
- **Aplikace Calling card**
- **Překlad čísel**
- **Prediktivní volič** (Predictive dialer)
- **Vzdálené „kanceláře“ pro existující PBX**
- **Řazení volání do front se vzdáleným zprostředkovatelem**

3.3.3 Použité technologie

Asterisk byl navrhován tak, aby podporoval používání nových rozhraní a umožnil snadné přidání nových technologií.

Tyto rozhraní jsou rozděleny do tří základních skupin:

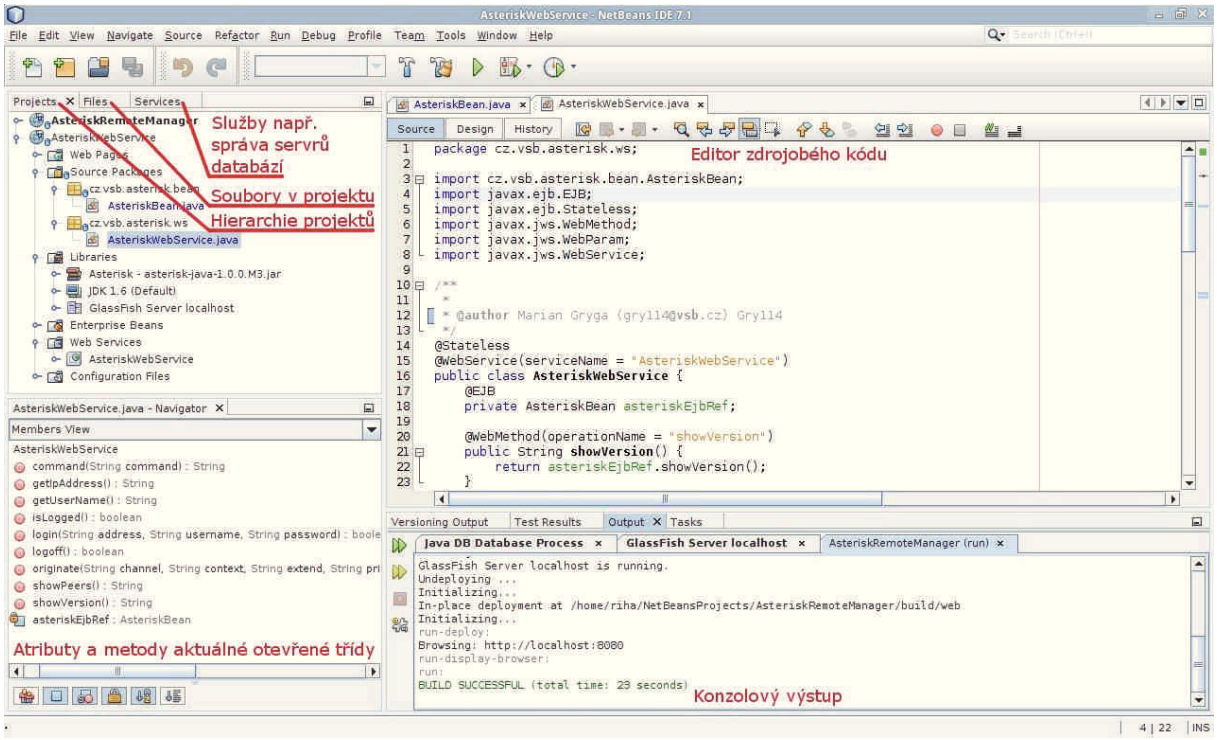
- **Zaptel rozhraní** - dodává firma Digium [08] pro různé varianty síťových rozhraní (např. PSTN, POTS, T1, E1 a mnoho dalších).
- **Non - Zaptel rozhraní** - rozhraní směrované k tradičním telekomunikačním službám (např. ISDN4Linux, OSS/Alsa a další).
- **Pocket voice** - rozhraní poskytující komunikaci přes VoIP (např. MGCP, SIP, IAX, H.323).

V této kapitole jsem Vás seznámil se stručným popisem telefonní ústřednou Asterisk. Více informací a podrobný popis Asterisk PBX naleznete zde [08].

3.4 Vývojové prostředí

Implementace webové služby bude v jazyce Java EE. Zvolil jsem si tedy pro svou vlastní implementaci open source vývojové prostředí NetBeans [09]. Toto prostředí v sobě integruje všechny nástroje pro vývoj webových služeb prostřednictvím JAX-WS a také nástroje pro vývoj klienta

webové služby. Pro testování implementace je v tomto prostředí integrován aplikační server GlassFish, na kterém můžeme bez nutnosti instalace a konfigurace aplikačního serveru námi implementovanou službu i klienta testovat.



Obrázek 3.1Vývojové prostředí NetBeans.

3.5 Balíček Asterisk Java

Pro komunikaci s Asterisk serverem je nutno použít Asterisk Java balíček. Tento balíček nabízí množinu Java tříd pro snadnou interakci s Asterisk PBX serverem. Asterisk Java je poskytován v souladu s podmínkami licence Apache, verze 2.0. Tento balíček vyžaduje JRE verze 1.6. Ve verzi používané v této práci je podporovaná verze Asterisk 1.6. Popis tohoto rozhraní naleznete zde [10].

Asterisk Java nabízí tyto Asterisk rozhraní:

- **Fast Asterisk Gateway Interface** (FastAGI)
- **Asterisk Manager Interface** (AMI)

V této práci pro komunikaci s Asterisk PBX serverem je použít pouze AMI interface.

3.6 Vlastní implementace

V kapitole 3.2 Analýza jsem se věnoval popisu implementace pouze obecně. V této kapitole se budu věnovat vlastní implementace i s ukázkami kódu z webové služba a klienta.

3.6.1 Implementace webové služby

Dle zadání bylo nutno navrhnout webovou službu tak, aby splňovala kriteria pro vzdálenou zprávu komunikačního systému. Bylo tedy nutné tuto webovou službu spojit s Asterisk PBX serverem pomocí Asterisk Java a zachovávat toto spojení aktivní. K tomuto účelu byl vytvořen a importován do projektu balíček Asterisk Java, který poskytuje potřebné třídy pro navázání spojení s Asterisk PBX serverem. Je však nutno implementovat dvě třídy, které se budou o tuto komunikaci starat a poskytovat klientu webové služby potřebné rozhraní. Aby si webová služba uchovávala stav daného spojení, bylo potřeba implementovat stavovou EJB třídu, která se bude starat o celou komunikaci z Asterisk PBX serverem.

Webová služba obsahuje tyto třídy:

- **AsteriskBean.java**
- **AsteriskWebService.java**

3.6.1.1 AsteriskBean.java

Tato třída se stará o celkovou komunikaci z Asterisk PBX serverem Pro komunikaci využívá balíček Asterisk Java. Jedná se o EJB, tedy o řízenou serverovou komponentu z Java EE API. Hlavním cílem teto třídy je udržovat stav spojení z Asterisk PBX serverem a tento stav propagovat webové službě. Třída je proto stavová `@Stateful` a označena anotací `@LocalBean` což znamená, že nemusí implementovat samotné rozhraní třídy a toho rozhraní je tvořeno veřejnými metodami tohoto EJB. V této třídě je použit balíček Asterisk Java pro spojení s Asterisk PBX serverem.

3.6.1.2 Ukázka s AsteriskBean.java

```
import org.asteriskjava.manager.*
import javax.ejb.Stateful;
import javax.ejb.LocalBean;

@Stateful
@LocalBean
public class AsteriskBean
{
     private ManagerConnection manager;
     private ManagerConnectionFactory factory;
...
}
```

V příkladu 3.6.1.2 vidíme námi implementovanou třídu pro komunikaci s AMI rozhraním, která má privátní instanční proměnné pro realizující spojeni s AMI.

3.6.1.3 Ukázka metody pro komunikaci s AMI

```
public String loginAndCommand(String address, String username,
String password, Stirng command)
{
try {
      ManagerConnectionFactory factory = new
            ManagerConnectionFactory(address, username, password);
      ManagerConnection manager = faktory.createManagerConnetion();
      manager.login();
      CommandAction commandAction = new
            CommandAction("core show version");
      ManagerResponse commnadResponse =
            manager.sendAction(commandAction, 3000);
      String result = commnadResponse.getAttribute("__result__");
      manager.logoff();
      return result;
      }catch(IllegalStateException ex) {
      ...
      }catch (IOException ex) {
      ...
      }catch (AuthenticationFailedException ex) {
      ...
      }catch (TimeoutException ex) {
      ...
      } catch (IllegalArgumentException ex) {
      ...
      }
}
```

V příkladu 3.6.1.3 je předvedeno, jak se pomocí tříd Asterisk Java můžeme připojit k AMI a provést jednoduchý příkaz, který vypíše verzi Asterisk PBX serveru. Tento příklad bude demonstrovat stejný princip připojení a provádění příkazu, který je použit ve zdrojových kódech této

práce a slouží ke spojení s Asterisk PBX serverem. V příkladu jsou uvedeny i výjimky, které můžou nastat pří komunikaci s tímto rozhraní.

3.6.1.4 Veřejné metody třídy

- **login(String address, String username, String password) : boolean –** metoda slouží k přihlášení k Asterisk PBX serveru.
- **logoff(): boolean** – metoda slouží k ohlášení z Asterisk PBX serveru.
- **command(String command) : String** – metoda pro provádění jednotlivých příkazů AMI.
- **originate(String chanel, String context, String extend, String priority,**
- **String timeout, String callerID, String async, String actionID) : String** – metoda umožňující spojování hovorů v Asterisk PBX serveru. V této metodě však stačí zadat kontext a extend pro uskutečnění hovoru, další parametry jsou nepovinné.
- **showPeers() : String** – metoda slouží k výpisu SIP klientů v Asterisk PBX serveru.
- **showVersion() : String** – metoda slouží k zobrazení verze Asterisk PBX serveru.
- **isLogged() : boolean** – metoda slouží k zobrazeni jestli je uživatel přihlášen k Asterisk PBX serveru..
- **getIPAddress() : String** – metoda slouží k zobrazení IP adresy Asterisk PBX serveru.
- **getUserName() : String** – metoda slouží k zobrazení uživatele přihlášeného k Asterisk PBX serveru.

3.6.1.5 Privátní metody třídy

- **parseCommand(String command) : String –** metoda slouží k odstranění posledních znaků, které Asterisk Java interface přidává nakonec vraceného řetězce.
- **parsePeers(): String** – metoda slouží k převodu bílých znaků, především mezer v řetězci na čárky pro snadný převod předaného řetězce na straně klienta do tabulky. Tato metoda byla navržena speciálně pro Asterisk 1.6, nemusí tedy v jiných verzích korektně fungovat.

3.6.1.6 AsteriskWebService.java

Třída uvedená v 3.6.1.1 tvoří základ pro implementaci třídy reprezentující JAX-WS webovou službu a přebírá všechny metody popsané v 3.6.1.4. Tato třída musí byt bezstavové EJB, neboli `@Stateless` právě pro zachování stavu přihlášení k AMI. Tato třída tvoří koncový bod webové služby a metody jsou názvy operací této webové služby. Z této třídy pak vývojové prostředí 3.4 automaticky vygeneruje WSDL dokument.

3.6.1.7 Ukázka z AsteriskWebService.java

```
@Stateless
```

```
@WebService(serviceName = "AsteriskWebService")
public class AsteriskWebService {
    @EJB
    private AsteriskBean asteriskEjbRef;

    @WebMethod(operationName = "showVersion")
    public String showVersion() {
        return asteriskEjbRef.showVersion();
    }
}
```

Příklad 3.6.1.7 ukazuje JAX-WS implementaci třídy webové služby a jednu z poskytovaných metod této služby. Vidíme zde i třídu z 3.6.1.1 označenou anotací `@EJB` a vidíme, že metoda v ukázce vrací přímo stejnou metodu EJB třídy.

3.6.2 Implementace klienta webové služby

Po samotné implementaci webové služby, bylo nutno implementovat klienta této služby, hlavně z důvodu snadného ovládání a vykonávaní jednotlivých příkazů AMI. Vývojové prostředí uvedené v 3.4 nabízí možnost importu WSDL dokumentu a automatické vygenerování rozhraní koncového bodu pomocí JAX-WS popsané v 2.3. Jednotlivé metody webové služby budou pak volány z Java Servlets a prezentovány uživateli pomocí JSP stránek. Bylo tedy nutno vytvořit více těchto Servlet tříd a k ním JSP pro většinu metod poskytované webovou službou. Pro lepší vzhled pak budou tyto JSP stránky nestylována pomocí CSS stylů.V klientu bylo však potřeba převést řetězec vracený metodou `command()` do HTML pro správné zobrazení v JSP stránce. Pro tento účel byl použit balíček Commons Lang [11] společnosti Apache a zejména třída StringEscapeUtils. Bylo nutno nahradit bíle znaky především (mezeru HTML) entitou ` `. Převod řetězce do HTML řeší metod `toHtml(String string)` ve třídě Command.java. Cílem teto práce však není popis technologie Java EE. Proto více informací o Java Servlet a JSP naleznete zde [12] a zde [13].

3.6.2.1 Servlety implementovány na straně klienta

- **Login.java –** Servlet sloužící k přihlášení klienta k Asterisk PBX serveru voláním metody `login()` webové služby.
- **Logoff.java** – Servlet sloužící k ohlášení z Asterisk PBX serveru voláním metody `logoff()` webové služby.

- **Command.java** – Servlet sloužící k provadění jednotlivých příkazů AMI voláním metody `command()` webové služby.
- **Originate.java** – Servlet sloužící pro spojování hovorů Asterisk PBX serveru voláním metody `originate()` webové služby.
- **ShowPeers.java** – Servlet sloužící k zobrazení SIP uživatelů v Asterisk PBX serveru voláním metody `showPeers()` webové služby.

3.6.2.2 JSP stránky implementovány na straně klienta

- **index.jsp** – JSP stránka sloužící k přihlášení klienta k Asterisk PBX serveru odesláním formuláře Servletu Login.java
- **error.jsp** – JSP stránka sloužící k zobrazení chyb během komunikace.
- **main.jsp** – Hlavní JSP stránka zobrazená po úspěšném přihlášení klienta, na této stránce je zobrazena aktuální verze Asterisk PBX serveru.
- **command.jsp** – JSP stránka sloužící k provádění jednotlivých příkazů AMI odesíláním formuláře Servetu Command.java.
- **originate.jsp** – JSP stránka sloužící pro spojování hovorů Asterisk PBX serveru odesíláním formuláře Servetu Originate.java.
- **showpeers.jsp** – JSP stránka sloužící pro zobrazení tabulky SIP uživatelů v Asterisk PBX.

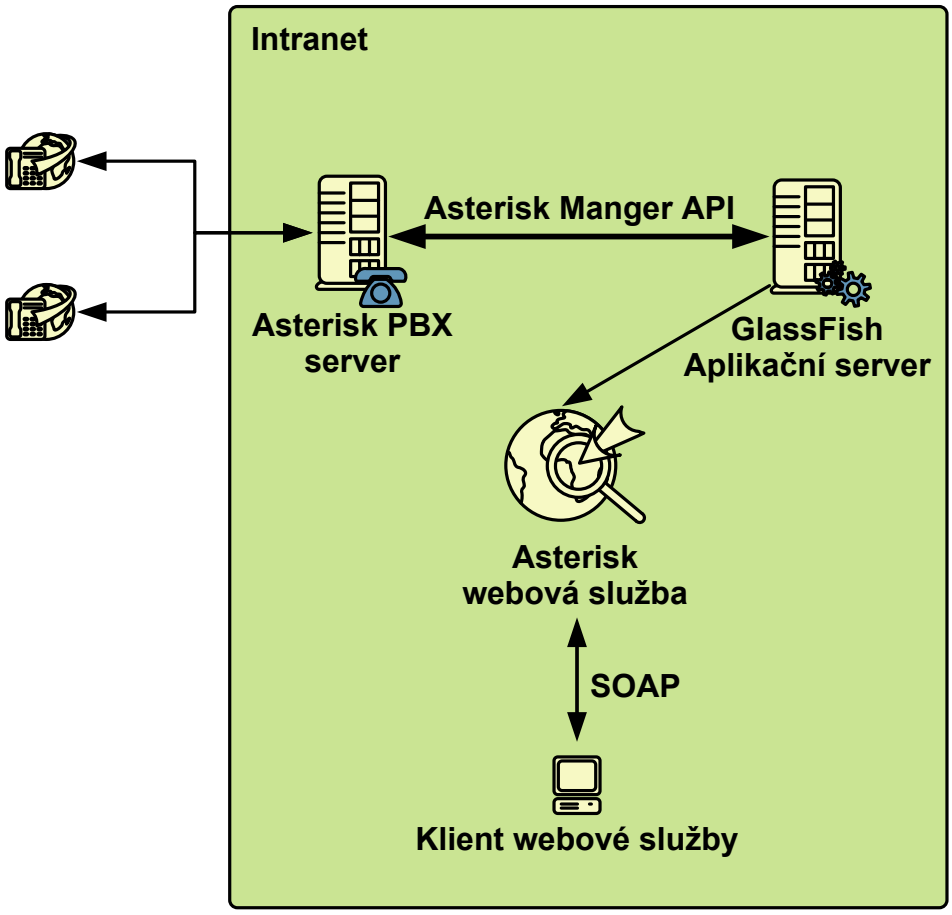
3.6.2.3 Ukázka volání webové služby metodou doPost v Servletu

```
@Override
protected void doPost(HttpServletRequest request,
HttpServletResponse response)throws ServletException, IOException {
     AsteriskWebService_Service service = new
          AsteriskWebService_Service();
     AsteriskWebService asteriskWS =
          service.getAsteriskWebServicePort();
     String commandResponse = toHtml(asteriskWS.command(command));
     request.setAttribute("commandresponse", commandResponse);
     request.getRequestDispatcher("pages/command.jsp").forward(reque
st, response);
}
```

Příklad uvedený v 3.6.2.3 ukazuje volání webové služby Servletem a následné předání odpovědi JSP stránce. Všechny zdrojové kódy uvedené v této kapitole jsou k dispozici v Přílohách na DVD.

4 Testování a ověřování funkčnosti s koncovými zařízeními

Testování celého systému bude probíhat především prostřednictvím klienta webové služby a nástroje soapUI. Webová služba i klient webové služby budou vedle sebe nasazeny na aplikační server GlassFish. Komunikační systém Asterisk bude nainstalován na druhém serveru. Prostředí serveru bude nasimulováno v Oracle Virtual Box na Linuxové distribuci Ubuntu. Na obrázku 4.1 vidíme architekturu tohoto systému.



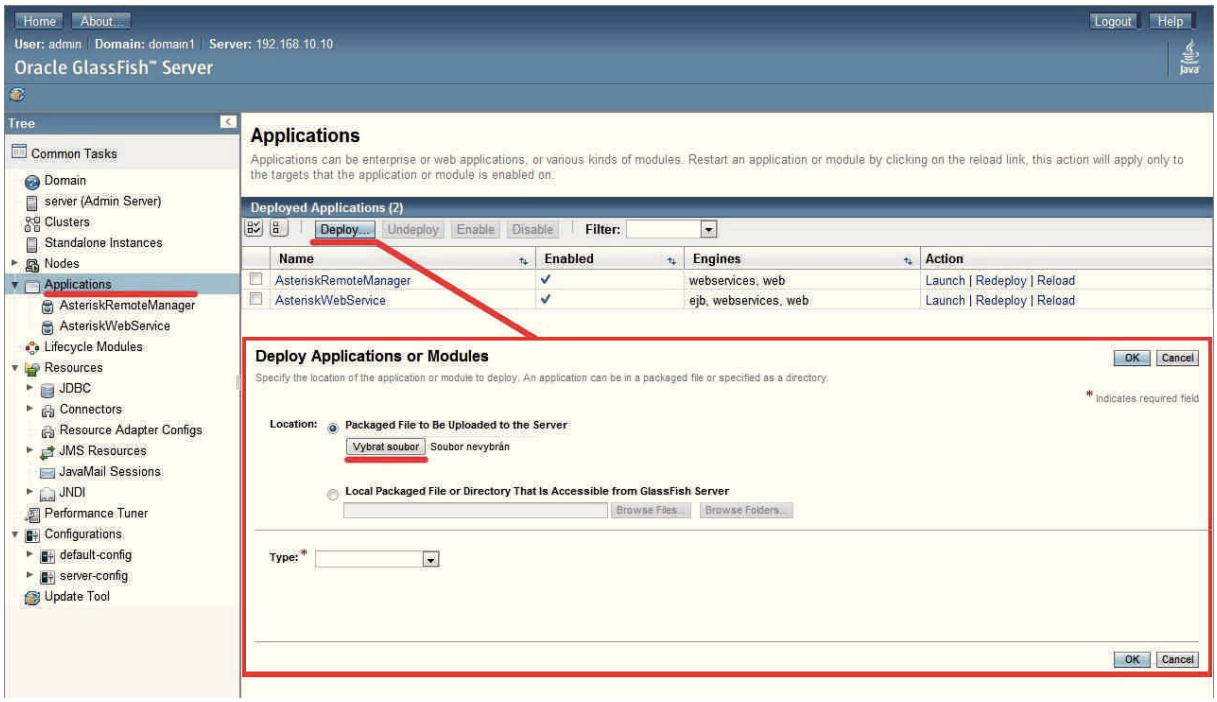
Obrázek 4.1 Architektura komunikačního systému

4.1 Instalace a konfigurace serverů v prostředí Virtual Box

Samotná instalace a konfigurace serverů Asterisk PBX a GlassFish nejsou tématem této práce. Na internetu však existuje spousty návodů jak oba tyto servery v Ubuntu distribuci nainstalovat a nakonfigurovat.

4.2 Nasazení webové služby a klienta webové služby na server GlassFish

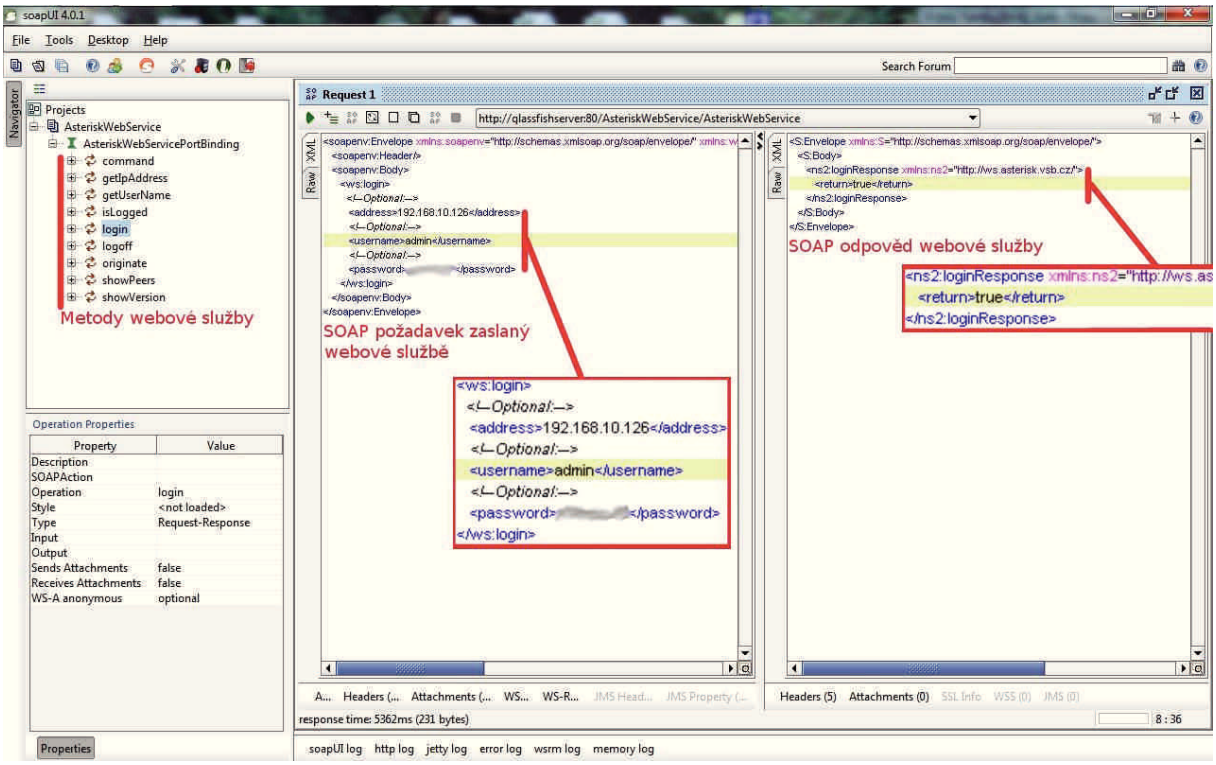
Na obrázku 4.3 je ukázka jakým způsobem je možno vlastní aplikace nasadit na aplikační server GlassFish. Prvním krokem pro nasazení na server je spuštění administrátorské konzole. To provedeme ve webovém prohlížeči zadáním adresy serveru např. 192.168.10.10 a portu 4848. Cele URL pro administrátorskou konzolu dle příkladu by mělo vypadat takto: 192.168.10.10:4848. Je potřeba se k serveru přihlásit, údaji nastavenými při instalaci serveru. Obrázek 4.2 ukazuje jak nasadit požadovanou aplikaci na GlassFish server.



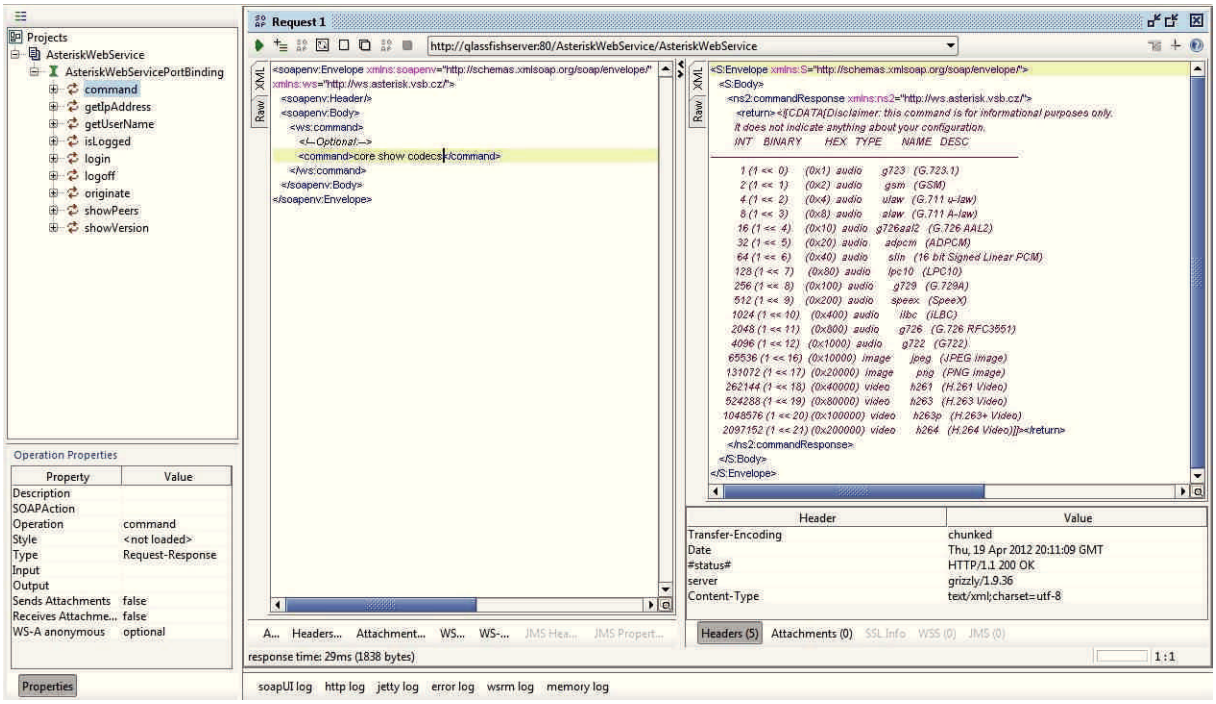
Obrázek 4.2 Nasazení na server GlassFish

4.3 Testováni webové služby pomocí soapUi

SoupUI je nástroj primárně určený k testování SOAP webových služeb. Proto použiju tento nastroj k otestování implementované webové služby. Na obrázku 4.3 vidíme prostředí SoapUI v němž lze po importu WSDL dokumentu webové služba volat příslušné metody této služby. Na tomto obrázku také vidíme dotaz a odpověď zaslanou webové službě pomocí SOAP zprávy. V tomto případě se jedná o metodu `login` s parametry address, username a password, která vrací true pokud bylo přihlášení úspěšné. Obrázek 4.4 ukazuje použití metody `command` v AMI a to konkrétně `core show codesc`, jehož návratová hodnota je řetězec obsahující výpis kodeků v Asterisk PBX serveru. Použité obrázky jsou pouze informativní, plnou velikost obrázků jsou uloženy v přílohách na DVD.



Obrázek 4.3 Prostředí soapUI



Obrázek 4.4 Prostředí soapUI

4.4 Testováni webové služby pomocí klienta

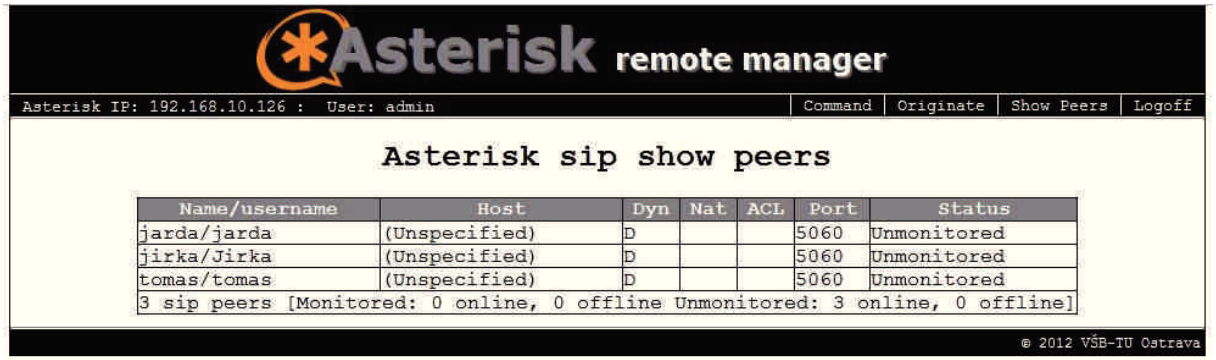
Pro ověření funkčnosti a otestování webové služby byl implementován klient této webové služby popsaná v 3.6.2. Tento klient používá několik obrazovek pro interakci z uživatelem o umožňuje uživateli ovládání Asterisk PBX serveru pomocí této webové služby jak již bylo zmíněno v předchozích kapitolách.

Obrázek 4.5 Přihlašovací obrazovka klienta

Obrázek 4.5 zobrazuje obrazovku pro přihlášení klienta webové služby k Asterisk PBX serveru. Pro přihlášení je nutno zadat adresu serveru, přihlašovací jméno a heslo.

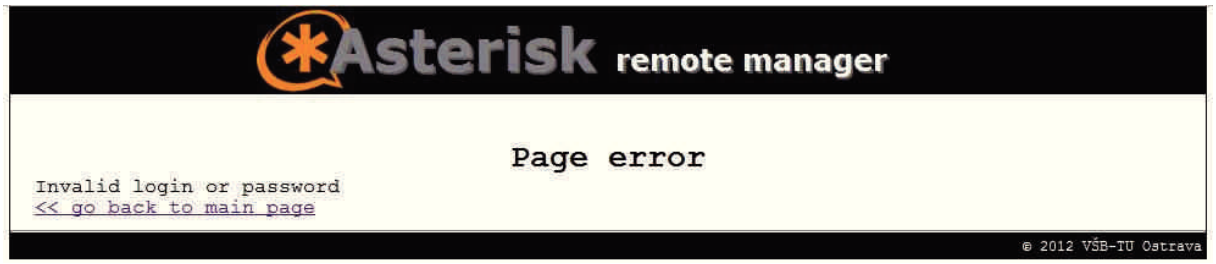
Obrázek 4.6 Úvodní obrazovka

Obrázek 4.6 ukazuje úvodní obrazovku po přihlášení klienta k Asterisk PBX serveru. Vlevo nahoře vidíme adresu serveru a přihlášeného uživatele. Vpravo pak jednotlivé odkazy na ostatní obrazovky, které nám klient nabízí.



Obrázek 4.7 Obrazovka s tabulkou SIP uživatelů

Obrázek 4.7 znázorňuje SIP uživatele na Asterisk PBX serveru. Zde můžeme sledovat kdo je přihlášen, jeho IP adresu a další údaje o uživateli.

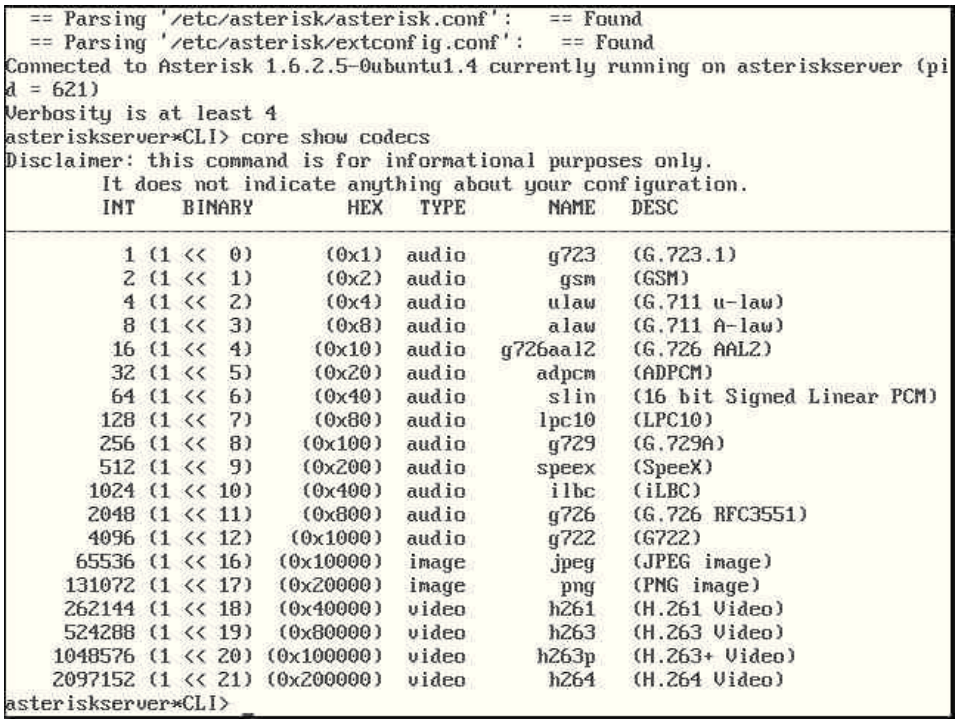


Obrázek 4.8 Obrazovka chyby

Na obrázku 4.8 vidíme obrazovku pro znázornění chyby klienta. Tato chyba může nastat v průběhu komunikace a je doplněná chybovou zprávou v tomto případě špatně zadaným uživatelským jménem nebo heslem. Konkrétní chyby jdou nalézt i v logu serveru GlassFish, kde jsou zapisovány při vyvolání výjimky.

Obrázek 4.9 Obrazovka pro vykonávaní příkazů AMI

Obrázek 4.9 demonstruje vykonávání příkazů AMI. Následující příklad ukazuje použití příkazu `core show codecs` , jenž jsem použil i při testování v soapUI. Pro výpis všech příkazu AMI slouží příkaz help, který nám napoví co všechno a jak můžeme pomocí tohoto klienta na Asterisk PBX serveru vykonávat. Pro demonstraci zde přidávám obrázek 4.10, který demonstruje sejný příkaz přímo na Asterisk PBX serveru.



Obrázek 4.10 Výpis příkazu na serveru

5 Závěr

Téma této bakalářské práce bylo velmi zajímavé a poučné. Webové služby jsou v dnešní době oblíbený nástroj pro vývoj aplikací v prostředí webu a hlavně prostředkem pro komunikaci více aplikací mezi sebou. Pří vypracovávání teoretické části práce jsem se dozvěděl mnoho informací nejen o webových službách, ale i o technologiích používaných v prostředí webu, které je možné využít i v praxi.

Pří vývoji webové služby se ukázalo, že nemusí vždy vše fungovat hned od začátku, tak jak by mělo. Jak se ukázalo nasazení nové verze aplikačního serveru všechny problémy vyřešilo. Aplikace tak funguje bez sebemenších problémů. Použitím EJB jako hlavní třídy pro komunikaci s Asterisk Java rozhraním odpadla nutnost přihlašování se pro každé vykonávaní příkazů a metod. Tato třída je připravena na další libovolné rozšíření a může tak sloužit jako příklad, jak používat AMI a poskytnout nástroj pro další rozšiřování této webové služby.

Klient webové služby slouží hlavně pro její ovládání a umožňuje komplexní vzdálenou zprávu komunikačního systému pomocí AMI příkazů. Klienta je možno zobrazit v dnes běžně používaných webových prohlížečích a poskytuje tak nástroj pro snadné ovládání implementované webové služby.

Testování prokázalo, že aplikace splňuje očekávání. Webová služba byla nasazena na aplikační server GlassFish v Linux prostředí. Samotný server GlassFish byl nenáročný na hardwarové požadavky a pracoval bez problémů i s 256MB RAM. Klient byl testován na Windows i Linux platformách na různých strojích.

Použitá literatura

[01] XML Schema Part 2: Datatypes Second Edition. In: *W3C* [online]. 28 October 2004 [cit. 2012-01-03]. Dostupné z: http://www.w3.org/TR/xmlschema-2/

[02] XML. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2012-01-05]. Dostupné z: http://en.wikipedia.org/wiki/Xml

[03] XML Tree. In: *W3Cschool* [online]. 2002 [cit. 2012-01-08]. Dostupné z: http://www.w3schools.com/xml/xml_tree.asp

[04] *Building Web services with Java*: *making sense of XML, SOAP, WSDL, and UDDI*. 2nd ed. Indiana: Sams Publishing, c2005, 792 s. ISBN 06-723-2641-8.

[05] KALIN, Martin. *Java web services*: *up and running*. 1st ed. Sebastopol, Calif.: O'Reilly, c2009, 297 s. ISBN 05-965-2112-X.

[06] ORACLE. *Java EE 5* [online]. 2007 [cit. 2012-01-12]. Dostupné z: http://docs.oracle.com/javaee/

[07] Asterisk manager API. In: *Voip-info.org* [online]. 2005 [cit. 2012-02-03]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+manager+API

[08] WINTERMEYER, Stefan a Stephen BOSCH. *Practical Asterisk 1.4 and 1.6*. Upper Saddle River: Addison-Wesley, 2009, 793 s. ISBN 978-0-321-52566-6.

[09] *NetBeans* [online]. 2000 [cit. 2012-02-06]. Dostupné z: http://netbeans.org/

[10] Asterisk Java [online]. 2006 [cit. 2012-03-08]. Dostupné z: https://blogs.reucon.com/asterisk-java/

[11] *Common Lang* [online]. 2001 [cit. 2012-03-09]. Dostupné z: http://commons.apache.org/lang/

[12] Servet. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2012-03-11]. Dostupné z: http://en.wikipedia.org/wiki/Java_Servlet

[13] JSP. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2012-03-11]. Dostupné z: http://en.wikipedia.org/wiki/JavaServer_Pages

Seznam příloh

Příloha A: Adresářová struktura přiloženého DVD

/Obrázky	Plná velikost obrázků pojmenovaných dle čísel v dokumentu.
/Webová služba	NetBeans projekt webové služby. Zdrojové kódy.
/Webová služba kódy	Zdrojové kódy služby samostatně.
/Klient	NetBeans projekt webové služby. Zdrojové kódy.
/Klient kódy	Zdrojové kódy klienta samostatně.
/Aplikace webová služba	Aplikace verze 1.0 ve formátu .war k nasazení na server GlassFish.
/Aplikace klient	Aplikace verze 1.0 ve formátu .war k nasazení na server GlassFish.